УДК 004.65:004.032.2 DOI: https://doi.org/10.30837/IYF.PCEIP.2024.045

ОСОБЛИВОСТІ ВИКОРИСТАННЯ APACHE KAFKA У РОЗПОДІЛЕНИХ СИСТЕМАХ РЕАЛЬНОГО ЧАСУ

Погорєлова Л.А.
Науковий керівник – к.т.н., доц. каф. КІТС, доц. Сердюк Н.М.
Харківський національний університет радіоелектроніки
(61166, Харків, просп. Науки, 14, каф. КІТС, тел.: (057)702-02-45)
e-mail: liliia.pohorielova@nure.ua

The article provides a detailed description of distributed systems for real-time data analysis and processing. In the modern world, where speed, accuracy, and timeliness of information are becoming key success factors, understanding, and using Big Data and data processing is extremely important. To achieve this goal, it is necessary to use appropriate tools and modern approaches that allow efficient processing, analysis, and use of data. Apache Kafka is one of such tools. It supplies useful methods of communication between producers and consumers. The basic principles of the message broker architecture were considered.

У сучасному світі кількість даних, що генеруються та накопичуються, зростає експоненційно. Великі обсяги даних (Big Data) стали нормою, а їх аналіз та обробка в реальному часі стає все більш важливою задачею для багатьох сфер діяльності. Високоефективні системи, що побудовані з використанням брокерів повідомлень, стають ключовим інструментом для вирішення цієї задачі.

На основі таких систем створюються сучасні потокові веб-додатки, головне призначення яких полягає в аналізі даних з різних джерел по мірі їх надходження. Такі програми дозволяють компаніям та організаціям швидко отримувати актуальну інформацію, реагувати на зміни в реальному часі та впроваджувати розумні аналітичні підходи для побудови стратегічно вигідніх рішень. Прикладами таких додатків є системи моніторингу та пропонування рекомендацій, фінансові системи, системи аналізу соціальних мереж, системи прогнозування та багато інших.

На сьогоднішній день існує велика кількість інструментів для обробки великих обсягів даних, серед яких до найпопулярніших належать: Apache Spark та Apache Hadoop. Перед безпосереднім аналізом інформації постає проблема доставки даних. Саме це завдання вирішується багатопотоковою платформою Apache Kafka. Це розподілена система передачі повідомлень з високою пропускною здатністю та низькими затримками. Вона здатна обробляти великі обсяги потокових даних з використанням структурованої архітектури журналів. Kafka надає надійну передачу повідомлень, розділення потоку даних на теми та можливість горизонтального масштабування.

Основу архітектури складає Kafka Cluster, що вміщує у собі набір брокерів, тем та розділів [1]. На рисунку 1 схематично представлено взаємозв'язок усіх компонентів.

Брокери (brokers) утворюють основну складову системи. Кожен брокер є незалежним сервером, який відповідає за зберігання та обробку повідомлень. Саме ці сутності отримують повідомлення від видавців (продюсерів) і відправляють їх підписникам (споживачам). Kafka Cluster може містити кілька брокерів, що дозволяє розподілити навантаження та забезпечити відмовостійкість.

Теми (topics) є категоріями або каналами, до яких видавці публікують повідомлення та на які підписуються споживачі. Вони представляють собою логічне розділення даних або потоків повідомлень. Кожна тема може мати декілька розділів, які розподіляють дані всередині теми між брокерами. Теми можуть бути реплікованими, що дозволяє зберігати копії даних на різних брокерах для забезпечення надійності та доступності.

Розділи (partitions) є фізичними одиницями зберігання даних в межах тем. Вони дозволяють розподіляти дані між брокерами та обробляти їх паралельно. Кожен розділ може мати свій власний набір повідомлень, які зберігаються у впорядкованому логу. Підписники можуть споживати повідомлення з різних розділів відповідно до своїх потреб.

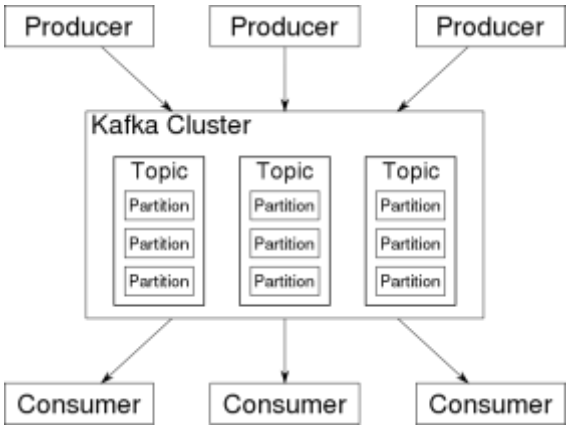

Рисунок 1 – Архітектура Apache Kafka

Прикладом реальної системи з використанням Apache Kafka може стати платформа моніторингу стану здоров'я пацієнтів [2]. Дана система направлена на аналіз та обробку великих і різнорідних даних, зібраних біомедичними датчиками задля полегшення процесів класифікації та діагностування захворювань медичним персоналом. Запропонована платформа складається з чотирьох рівнів: моніторинг пацієнтів у реальному часі, прийняття рішень і зберігання даних у реальному часі, класифікація пацієнтів і діагностика захворювань, а також пошук і візуалізація даних.

На першому етапі відбувається збір даних з різних джерел та їх потокова обробка. Дані з активних біосенсорів, які дозволяють безперервно контролювати стан пацієнтів, надсилаються в режимі реального часу до

сховища даних за допомогою системи обміну повідомленнями Apache Kafka.

Другий етап передбачає використання Spark і Hadoop HDFS (розподілена файлова система Hadoop) для аналізу та зберігання даних відповідно. Після встановлення на головному вузлі кластера Spark отримує дані, що надходять від Kafka, і застосовує алгоритми виявлення надзвичайних ситуацій та пошуку відсутніх записів перед відправкою остаточних даних до HDFS для зберігання.

На третьому етапі здійснюється класифікація пацієнтів і діагностика захворювання з використанням пакетної обробки та пошуком кореляції після збереження даних. Останній етап присвячений отриманню та візуалізації оброблених даних за допомогою модулю SparkSQL.

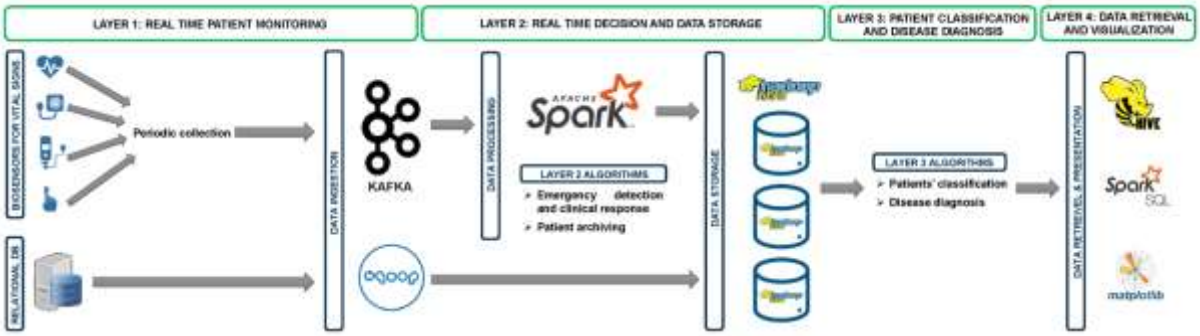

Рисунок 2 – Приклад архітектури системи обробки Big Data

Таким чином, Apache Kafka відіграє першочергову роль системах обробки великих даних, забезпечуючи надійний та ефективний потік даних в реальному часі. Архітектура розглянутого брокеру повідомлень забезпечує зручну інтеграцію з іншими компонентами екосистеми Hadoop, такими як Spark і HDFS, що дозволяє легко обробляти, зберігати і аналізувати великі обсяги даних в розподіленому середовищі.

Список використаних джерел

1. Levy E. Kafka vs. RabbitMQ: Architecture, Performance & Use Cases Blog Upsolver, 2019. URL: https://www.upsolver.com/blog/kafka-versus-rabbitmq-architecture-performance-use-case (дата звернення 03.03.2024).

2. Harb H., Mroue H., Mansour A. Hadoop-Based Platform for Patient Classification and Disease Diagnosis in Healthcare Applications, 2020. URL: https://www.mdpi.com/1424-8220/20/7/1931 (дата звернення 03.03.2024).